НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: Комплекс задач для аналізу та прогнозування потреб споживачів

Виконав: студент *IV* курсу, групи *ІС-61*

Кравцов Мирослав Віталійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.ф.-м.н. Клименко Олена Миколаївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Тамара Олексіївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц., к.т.н., доц. Лісовиченко Олег Іванович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року
Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Кравцову Мирославу Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Комплекс задач для аналізу та прогнозування потреб споживачів* »

керівник проєкту *Клименко Олена Миколаївна к.ф.-м.н.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів
5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна класів програмного забезпечення

2. Схема структурна послідовності

3. Схема структурна компонентів програмного забезпечення

4. Схема структурна активності

5. Схема структурна варіантів використань

6. Схема бази даних

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема структурна послідовності

3. Схема структурна класів програмного забезпечення

4. Схема структурна розгортання програмного забезпечення

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*		
2.	*Аналіз існуючих методів розв'язання задачі*		
3.	*Постановка та формалізація задачі*		
4.	*Розробка інформаційного забезпечення*		
5.	*Алгоритмізація задачі*		
6.	*Обґрунтування використовуваних технічних засобів*		
7.	*Розробка програмного забезпечення*		
8.	*Налагодження програми*		
9.	*Виконання графічних документів*		
10.	*Оформлення пояснювальної записки*		
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Мирослав КРАВЦОВ

Керівник Олена КЛИМЕНКО

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6114.000 ПЗ	Пояснювальна записка	71	
3	А4	ДП 6114.01.000 ТЗ	Технічне завдання	15	
4	А3	ДП 6114.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6114.03.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
6	А3	ДП 6114.04.000 ССР	Схема структурна розгортання	1	
			програмного забезпечення		
7	А3	ДП 6114.05.000 ССП	Схема структурна послідовності	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6114 00.000.00		
Розробн.		Кравцов М.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Клименко О.М					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-61	
Н/контр.		Тєлишева Т.О.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Комплекс задач для аналізу та прогнозування потреб споживачів

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з 5 розділів, містить 14 рисунків, 5 таблиць, 1 додаток та 11 джерел.

Дипломний проєкт присвячений розробці комплексу задач для аналізу клієнтської бази, та передбаченні поведінки покупців.

Призначенням розробки є автоматизація кластерного аналізу клієнтської бази, та отримання графічного матеріалу для допомоги у аналізі поведінки клієнтів.

У розділі загальних положень описані функціональні вимоги до системи, визначені цілі та призначення розробки, сформульована постановка задачі, та проаналізовані існуючи аналоги.

У розділі інформаційного забезпечення надано детальний опис вхідних та вихідних даних, разом із структурою даних та описом атрибутів.

У розділі математичного забезпечення були сформовані математична та змістовна постановки задачі, та описані використані алгоритми.

Розділ програмного забезпечення описує засоби розробки програмного продукту та етапи проєктування його архітектури.

У технологічному розділі визначено мету проведення випробувань програмного продукту та описано їх результати.

КЛАСИФІКАЦІЯ, КЛАСТЕРНИЙ АНАЛІЗ, КЛІЄНТСЬКА БАЗА, АЛГОРИТМИ МАШИННОГО НАВЧАННЯ, ВПОДОБАННЯ КОРИСТУВАЧА.

ABSTRACT

The structure and the scope of the work. *Explanatory note of the diploma project consists of 5 sections, contains 14 drawings, 5 tables, 1 application and 11 sources.*

The diploma project is devoted to the development of a software solutions for analysis and forecasting of consumer needs.

The purpose of the development is to automate the cluster analysis of the customer base, and to obtain graphical material to help analyze customer needs.

The section of general provisions describes the functional requirements for the system, defines the goals and purposes of development, formulates the problem statement, and analyzes the existing analogues.

The information support section provides a detailed description of the input and output data, along with a data structure and attribute description.

In the section of mathematical support the mathematical and substantial statement of the problem were formed, and the used algorithms were described.

The software section describes the software development tools and design steps for its architecture.

The technological section defines the purpose of software product testing and describes their results.

CLASSIFICATION, CLUSTER ANALYSIS, CUSTOMER BASE, MACHINE LEARNING ALGORITHMS, USER'S LIKE.

ЗМІСТ

ВСТУП

У наш час зростаючої популярності інтернет мережі майже кожен підприємець має свій он-лайн магазин, а ті хто не мають поспішають їх відкрити. Під час карантину потреба мати інтернет магазин стала очевидною для кожного продавця. Інтернет магазини не тільки дають змогу покупцям зручно зробити покупку із будь-якої точки планети, вони ще й економлять купу грошей. Власники інтернет магазинів витрачають менше грошей на рекламу, тримають мінімальну кількість персоналу, та не платять за оренду приміщення. Більш того, інетрнет магазин дає ще одну перевагу своїм власникам – збір інформації. Історія покупок користувача, його кліки, середні витрати за 1 чек, його країна, вік та стать – усе це може стати корисним для розвитку вашого магазину. Велекі магазини вже за декілька місяців роботи можуть назбирати гігабайти даних. На перший погляд ці дані можуть показатися неінформативними, проте це в корні невірно, грамотний спеціаліст та декілька алгоритмів машинного навчання можуть перетворити цю купу інформації на великі гроші.

Даний дипломний проєкт присвячений кластерному аналізу клієнтської бази, та перевірці наскільки корисним буде аналіз його роботи.

Кластерний аналіз – це технологія, що дозволяє розподілити вхідні дані на відносно однорідні класи.

Машинне навчаня – це підгалузь інформатики, у якій комп’ютер «навчається» виконувати складні задачі без явного програмування.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

У наш час кожна торговельна мережа має інтернет магазина, у яких кожен день люди витрачають купу грошей. Кожен цей магазин має базу даних, де фіксується кожне ваше придбання. У великих компаніях, які вже багато років продають свій товар он-лайн, існують величезні сервери, на яких зберігається уся інформація. Правильно використовуючи ці дані можна суттєво збільшити прибуток своєї компанії, та краще зрозуміти своїх клієнтів. Цю задачу неефективно вирішувати за допомогою звичайних аналітиків, через те що при великій кількості клієнтів робота займе дуже багато часу, більш того при аналізі поведінки споживачів аналітики мають звертати увагу на особистість покупця, що не є дуже етичним з точки зору конфіденційності клієнтів. З цими проблемами може допомогти алгоритми машинного навчання, а саме кластерний аналіз.

Кластеризація – це метод пошуку закономірностей, призначений для розбиття сукупностей об’єктів на однорідні групи або пошуку існуючих взаємозв’язків у даних. Ціллю кластерного аналізу є отримання нових знань, з допомогою яких в нашому випадку користувач зможе підлаштуватися під кожного свого клієнта, а не працювати із усіма однаково.

1.2 Аналіз існуючих аналогів

Задача кластеризації клієнтської бази даних має консалтинговий характер та потребує пошуку індивідуального підходу, тобто складно повністю автоматизувати процес. Проте є декілька програм, які виконують схожі функції.

Carrot Quest – сайт, який автоматизує деякі функції маркетингу. Даний ресурс надає багато послуг, серед яких: аналіз поведінки ваших покупців, автоматизація воронки, Web-Push сповіщення та сегментація клієнтів. З нашою роботою перетинається лише остання послуга. Carrot Quest пропонує інструмент для сегментації клієнтів, який розділить їх за наступними ознаками:

- наявність контактних даних;
- географічне знаходження;
- активність за останній час;
- канал-джерело ліда (таргетована реклама, email розсилання та інш.).

Серед плюсів Carrot Quest можна відзначити зручний інтерфейс, та широкий спектр послуг. До мінусів можна віднести високі тарифи (від 1000 до 85000 рублів за місяць), відсутність порівняння алгоритмів кластеризації та відсутність 100 відсоткової автоматизації (деякі послуги виконуються експертами). Окрім Carrot Quest існують ще декілька сайтів із схожим функціоналом MindBox, Convead.io. Які мають схожий функціонал та дешевші.

У даній роботі буде проведений більш детальний кластерний аналіз, порівнянні алгоритми кластеризації, досліджений вплив метрик на результат кластеризації, надані графіки та діаграми згідно отриманим кластерам. До плюсів нашої реалізації також можна віднести те, що дана програма буде безкоштовною. Серед мінусів відносно існуючих аналогів можна виділити гірший інтерфейс застосунку, та високі вимоги до підготовки користувача (розуміння предметної області, вміння завантажити дані у необхідному форматі).

1.3 Опис функціольної моделі

Основним актором у системі є користувач. Для нього доступні функції завантаження бази даних, налаштування алгоритмів кластеризації та їх параметрів. Після цього користувач може застосувати кластерний аналіз та

переглянути його результати включаючи діаграми та графіки, які допоможуть краще зрозуміти поведінку своїх клієнтів.

Бажано щоб користувач гарно розумів предметну область, та мав змогу проаналізувавши графіки та діаграми знайти змістовні інсайти.

На рисунку 1.1 зображено діаграму варіантів використань.

Рисунок 1.1 Діаграма варіантів використань

1.4 Постановка задачі

1.4.1 Призначення розробки

Призначенням розробки є автоматизація кластерного аналізу клієнтської бази, та отримання графічного матеріалу для допомоги у аналізі поведінкі клієнтів. У наслідку сегментації користувач може знайти особистий підхід до кожного клієнта, що збільшить їх лояльність та позитивно відобразиться на прибутку.

1.4.2 Цілі та задачі розробки

Ціль розробки:

- покращення аналізу товарообігу та поведінки споживачів;
- проведення сегментації і кластеризації клієнтської бази;
- збільшення зацікавлення споживачів супутніми товарами;
- передбачення наступної покупки користувача.

Задачі розробки:

- проаналізувати існуючі додатки та визначити їх недоліки;
- порівняти алгоритми кластеризації, та вплив параметрів на їх; результат, зробити висновки про їх використання;
- розробити додаток який буде універсальним для будь-якої предметної області та легкий у освоєнні.

Висновок до розділу

У даному розділі ми описали функціональну модель та предметне середовище, сформували постановку задачі, визначили цілі, призначення, та задачі розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

У даній роботі використовуються алгоритми машинного навчання. Результат розв’язання такої задачі залежить від якості, та об’єму даних, на яких відбувається навчання алгоритмів. Тобто перед розробкою алгоритмів треба переконатися, що вибірка, яка використовується є інформативною, ті її об’єм відповідає вимогам поставленій задачі. Наше програмне забезпечення буде працювати із даними які включають в себе торгові транзакції реальних клієнтів.

Набір даних який відповідає поставленим вимогам був знайдений на сайті archive.ics.uci.edu. Цей файл містить інформацію про транзакції здійснені більш ніж 4000 користувачів із різних країн, протягом 2011 року. Покупки відбувалися у он-лайн магазині, який займається продажем унікальних подарунків на будь-який випадок життя. Багато клієнтів компанії є оптовими торговцями. Набір даних містить інформацію про 541000 транзакцій. Кожна транзакція має 8 атрибутів :

- номер транзакції (InvoiceNo);
- код товару (StockCode);
- опис товару (Description);
- кількість копій цього товару в чеку (Quantity);
- дата покупки (InvoiceDate);
- ціна товару (UnitPrice);
- id покупця (CustomerID);
- країна покупця (Country).

Як можна побачити дані атрибути – універсальні. Таку інформацію може зібрати будь-який власник он-лайн магазину. Перші 10 транзакцій можна побачити на рисунку 2.1.

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047.0	United Kingdom

Рисунок 2.1 – Формат даних

2.2 Вхідні дані

Реалізоване програмне забезпечення допомагає користувачу провести аналіз його ринку та передбачити поведінку його клієнтів. Іншими словами призначення ПЗ має консалтинговий характер. Результатом роботи такої програми є графічний матеріал, який містить інформацію про клієнтську базу користувача та результат кластерного аналізу. Наприклад, на рисунку 2.2 можна побачити інформацію про розподіл покупок відносно країн, із яких вони були зроблені.

Рисунок 2.2 Кількість покупок із кожної країни

Рисунок 2.3 показує середню скільки умовних одиниць в середньому витрачає покупець.

Рисунок 2.3 Середня сума покупки (£) .

На рисунку 2.4 можна побачити результат кластеризації товарів. Були проаналізовані назви усіх товарів, які були продані за весь проміжок часу. Потім ми почистили назви товарів від даних, які не несуть в собі важливої інформації (наприклад кольори), та на основі отриманих ключових слів, та історії покупок відповідних товарів провели кластерний аналіз, поділивши товари на 5 груп. Ці групи можна побачити на наступному рисунку.

Рисунок 2.4 Результат кластеризації товарів

Висновок до розділу

У даному розділі ми описали структуру вхідних даних, навели приклад вихідних даних, навели опис зміних, які містяться у вхідних даних.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

В основі системи, розробленої під час роботи над даним проєктом лежить модель, натренована на даних, які містять інформацію про покупки в он-лайн магазині. Алгоритм роботи програмного забезпечення має наступні кроки:

- розбити товари на 5 категорій;
- класифікувати клієнтів на 11 категорій ;
- коректно визначити категорію майбутніх клієнтів;
- передбачити поведінку користувача (кількість майбутніх покупок, які зробить покупець, та кількість його візитів на сайт).

Дана система не буде працювати як звичайна програма заснована на правилах (If-then). Великі об'єми, та різноманіття вхідних даних (кількість товарів, унікальна поведінка кожного користувача, та інш.) не дають можливості розробити увесь потрібний нам функціонал явно. Для таких задач доцільно використовувати алгоритми машинного навчання. При розробці системи була поставлена задача розробити модель, яка буде якісно обробляти вхідні дані, обрати алгоритми які мають реалізувати усі кроки алгоритму, та проаналізувати якість їх роботи. Алгоритм роботи цих методів, та їх результати будуть наведені у наступних розділах.

3.2 Математична постановка задачі

Маючи дані про покупки, зроблені у он-лайн магазині на протязі 12 місяців можна сформувати наступну задачу: розробити інформаційну систему, в основі якої буде лежати модель, натренована на вхідних даних. Ця модель має найкращим можливим способом передбачати поведінку нових клієнтів, та мати змогу перенавчатися при оновлені бази даних.

3.3 Обґрунтування методу розв’язання

Обґрунтування методу розв’язання буде поділене на декілька частин для кожного кроку роботи нашої програми.

3.3.1 Розбиття продуктів на категорії

Вхідні дані містять змінну Description, у якій надано короткий опис товару. Модель напряму не може використовувати цю інформацію. На основі цього описа 4000 унікальних товарів були поділені на 5 категорій. Обрати правильну кількість категорій допомогла silhouette score. Це метрика реалізована у бібліотеці sklearn, вона була порахована для кількості кластерів від 3 до 10. Silhouette score повертає значення від -1 до 1 на основі обрахованої відстані між кластерами. На рисунку 3.1 можна побачити ці значення для різних кількостей кластерів.

```
For n_clusters = 3 The average silhouette_score is : 0.10071681758064248
For n_clusters = 4 The average silhouette_score is : 0.12208239761153944
For n_clusters = 5 The average silhouette_score is : 0.1470081849157512
For n_clusters = 6 The average silhouette_score is : 0.14389841472426354
For n_clusters = 7 The average silhouette_score is : 0.15212220110144017
For n_clusters = 8 The average silhouette_score is : 0.1558201267218184
For n_clusters = 9 The average silhouette_score is : 0.11656173409117862
```

Рисунок 3.1 Порівняння кількості кластерів продуктів

Як можна побачити найкращі результати були для випадку, коли кількість кластерів дорівнює 8. У випадку з великою кількість кластерів (більшою за 5) інсують групи товарів які містять малу кількість елементів. Було вирішено проводити класифікацію на 5 кластерів. Кластерізацію виконали методом kmeans.

3.3.2 Кластерізація покупців

Кластеризація покупців – найважливіша задача нашого програмного забезпечення. При розподілі покупців ми звертали увагу на такі їх характеристики:

- середня сума чеку;
- процент категорій товарів серед усіх його покупок.

На основі даних цих зміних була проведена кластеризація покупців на 11 кластерів. Для її реалізації були використані 6 алгоритмів:

- Suport Vector Machine;
- Logistic Regression;
- K-neares Neighbours;
- Decision Tree;

- Random Forest;
- Gradient Boosting.

Перші 10 місяців наших даних будуть використані для тренування цих алгоритмів, та останні 2 для їх тестування. Після тестування була порівняна точність(precision) їх роботи, та обран найкращії алгоритм.
Усі 11 кластерів можна побачити на рисунку 3.2. та рисунку 3.3.

Рисунок 3.2 Кластери покупців (частина 1)

Рисунок 3.3 Кластери покупців (частина 2)

На рисунку 3.4 показано порівняння роботи алгоритмів.

```
Support Vector Machine
Precision: 65.93 %

------------------------------
Logostic Regression
Precision: 71.34 %

------------------------------
k-Nearest Neighbors
Precision: 67.58 %

------------------------------
Decision Tree
Precision: 71.38 %

------------------------------
Random Forest
Precision: 75.38 %

------------------------------
Gradient Boosting
Precision: 75.23 %
```

Рисунок 3.4 Порівняння точності алгоритмів

Як можна побачити алгоритм Random Forest правильно класифікував 75.38% даних. Саме цей алгоритм буде використовуватися для цього набору даних.

3.4 Опис методу розв’язання

3.4.1 K-means

Мабуть найпростіший алгоритм кластеризації. Його мета - мінімізувати суму середньоквадратичного відхилення кожної точки від її кластера, яка рахується за формулою 3.1 .

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2 \quad (3.1)$$

де S_i – і-тий кластер, k - кількість кластерів, μ_i - центр і-го кластеру. На кожній ітерації перераховується центр ваг для кожного кластеру, після чого усі точки на площині діляться на кластери(для кожної точки обирається найближчий центр ваг).

На рисунку 3.4 можна побачити результат роботи алгоритму для двох кластерів. Ромбом позначені центроїди кластерів, а кольоровими кругами елементи цих кластерів.

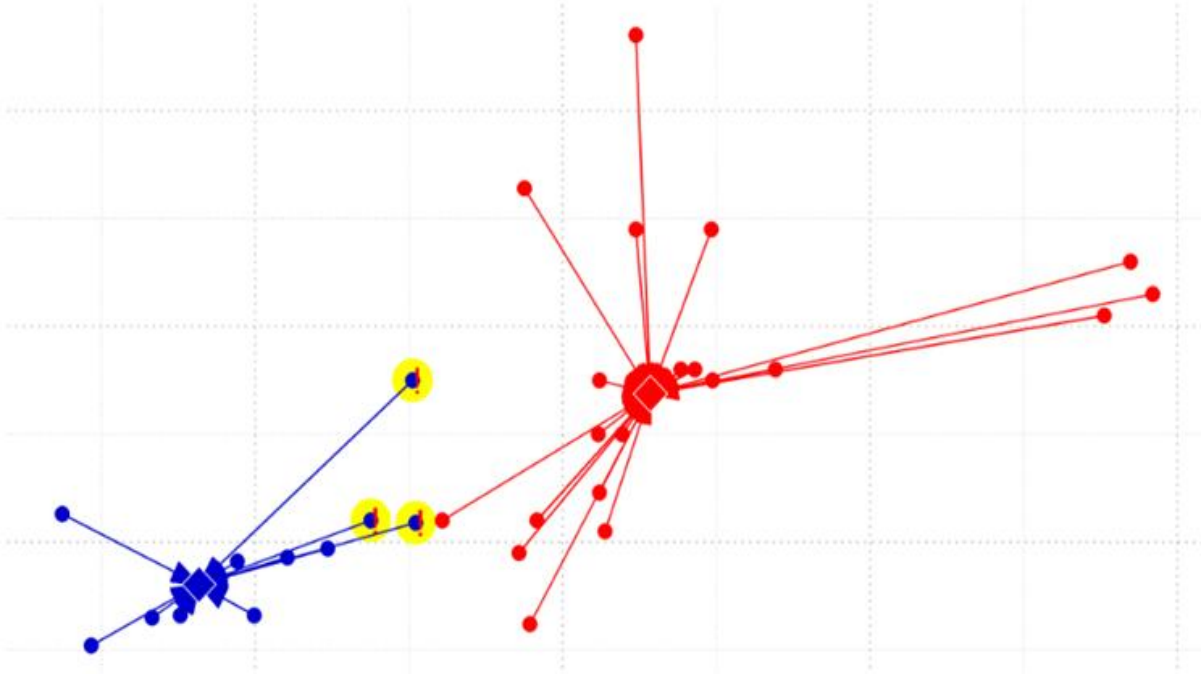

Рисунок 3.4 Візуалізація k-means

Цей алгоритм має наступні недоліки: необхідно самому задати кількість кластерів, результат роботи алгоритму дуже залежить від вибору початкових центрів ваг, погано працює із об’єктами що належать обом кластерам або не належать жодному. Останню проблему вирішує модифікована версія алгоритму – c-means. Результатом роботи алгоритму є матриця розміру n*k, де n – кількість всіх користувачів. Кожному користувачу у відповідність ставиться ймовірність що він належить кожному кластеру до кожного кластеру.

3.4.2 Logistic Regression

Логістична регресія може бути лінійною чи поліноміальною. У лінійній регресії результуюча змінна має два значення (елементи поділяються на два класи). У цьому випадку функція логістичної регресії повертає ймовірність належності елемента до одного з класів. Наприклад якщо маємо два класи "+" та "-", то як результат роботи для і-того елемента буде P_i^+. Значення P_i^- відповідно дорівнює 1 - P_i^+ .

В нашому випадку результуюча зміна має 11 значень. Отже задача відноситься до MultiClass classification. У таблиці 3.1 наведемо приклад вхідних даних для і-го покупця.

Таблиця 3.1 – Приклад вхідних даних для класифікації

№ покупця	Середня сума чеку	Відсоток j-го товару серез усіх покупок, $j = \overline{1,5}$				
		1	2	3	4	5
I	x_1= 57.25	x_2=15,5	x_3=10	x_3=35,5	x_4= 29	x_6 = 10

Як можна побачити по таблиці, в ході роботи алгоритма дані будуть представлені у 6-вимірному просторі. В основі роботи логістичної регресії лежить логістична функція, або сігмоїда. Функція сігмоїди рахується :

$$\text{Sigm(z)} = \frac{1}{1+e\text{^}z} \tag{3.2}$$

В якості аргумента сігмоїда приймає функцію f() від вектора вхідних даних x (3.3).

$$f(x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_6 x_{i6} \tag{3.3}$$

де β – вектор коефіцієнтів логістичної регресії, x_i – вектор вхідних даних для і-го покупця 3.1. Область визначення сігмоїди змінюється від -∞ до ∞, а область значення від 0 до 1. Ціль навчання моделі логістичної регресії – знайти найкращі коефіцієнти β. Функція активації для багато класової логістичної регресії показана у формулі 3.4:

$$P(x, y_i) = \frac{e^{f(x_i)}}{\sum_{j=1}^{k} e_j^{f(x_i)}} \tag{3.4}$$

де x це матриця яка містить інформацію про кожного покупця, k – кількість класів. Отримані ймовірності отримані на цьому моменті не будуть відповідати реальній ситуації. Модель потрібно навчити, це можна зробити мінімізувати функцію витрат J, яка рахується як показано на функціях 3.5 та 3.6.

$$J = \frac{1}{n}\sum_{i=1}^{n} H(T_i, P_i) \tag{3.5}$$

$$H(T_i, O_i) = -\sum_m T_i * \log(P_i) \tag{3.6}$$

де T_i це реальні значення класу і-го елемента у форматі вектору 1x11 заповненого нулями та одиницею у відповідній позиції, P_i це ймовірності

пораховані за допомогою формули вище, n – кількість усіх елементів у навчальній виборці. Мінімізацію функції витрат не можна виконати аналітично, треба використовувати методи оптимізації, такі як градієнтний спуск. Суть цього алгоритму полягає у оновлені матриці вагів за наступною формулою (3.7).

$$w_i := w_i - \Delta w_i * J(W) \quad (3.7)$$

де Δ це похідна.

Висновок до розділу

У третєму розділі нами були сформульовані змістовна, та математична постановки задачі, описані кроки роботи нашої системи, та наведений короткий опис деяких використаних алгоритмів.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Функціональна частина нашої системи містить алгоритми машинного навчання. Для реалізації систем такого типу найчастіше використовують одну із трьох мов: C++, R або Python. Серед сильних сторін C++ можна виділити поєднання можливостей як високорівневих мов програмування так і низькорівневих, можливість реалізації трудомістких програм та відносно швидке їх розв'язання. Серед мінусів використання цієї мови слід зазначити складність у вивченні, та необхідність писати велику кількість складного коду. Мова програмування R доволі часто застосовується у аналізі даних, та використовується для розв'язання задач машинного навчання (подалі МН). R часто використовується для візуалізації даних, та розробки таких задач МН як класифікація, регресія, та формування дерева рішень. Python – найпопулярніша із мов програмування, які використовуються у МН. Ця високорівнева мова програмування має багато інструментів для роботи із наукою о даних (Data Science), та технологією великих даних (Big Data).. Для розробки нашої системи була обрана мова Python 3. Серед переваг цієї мови можна виділити наступні:

Python3 – одна із найпростіших мов для вивчення. та використиння. У порівняння із реалізаціями на інших мовах програмування, реалізація задач на python зазвичай містить менше строк коду, та її легше читати.Писати код на мові Python швидше ніж на інших мовах програмування. Ця швидкість є дуже важливою у роботі із великими об'ємами даних. Python 3 має мабуть найкращу візуалізацію даних. Важлива частина роботи спеціалістів із сфери Data Science це розуміння предметної області, та вхідних даних. Python маж зручні і потужні бібліотеки для візуалізації даних. Серед них можна виділити ggplot, matplotlib, seaborn, NetwotkX, та Plotly. Приклад роботи цих бібліотек можна побачити у розділі 2.

Python 3 має дуже велику спільноту. Саме вона стоїть за розвитком мови, та створенням нових бібліотек. Якщо спеціаліст зустрічається із складною проблемою, він може попросити допомоги у спільноти на форумах

чи спеціальних сайтах. Більшість проблем із якими ви стикаєтеся вже були кимось вирішені.

Слабкою стороною Python можна назвати складність відстеження помилок у коді. Проблему повільного виконання застосувань на мові Python було вирішено із виходом платформи Anaconda.

Для обрахувань необхідних для роботи алгоритмів була використана бібліотека Numpy. Numpy – це open-source модуль Python, який надає загальні математичні та числові операції. Цей модуль надає можливість маніпулювати векторами та матрицями великих розмірів. Numpy написаний на мові C, через що усі обрахунки проводяться швидше, ніж якби, були просто написані на мові Python.

Для роботи із вхідними даними використовується бібліотека Pandas. Pandas - це високорівнева бібліотека, основою для якої служить бібліотека Numpy, що гарно впливає на продуктивність. Основними структурами даних у Pandas є Series (аналог асоціативного масиву) та DataFrames. Для роботи із базами даних у Pandas використовуються DataFrames. У них інформація зберігається у вигляді таблиць.

4.2 Загальні вимоги до технічного забезпечення

Дана система є веб-застосуваннєм, яке розгорнуте на Google Colab. Google Colab – це безкоштовний хмарний сервер який працює на основі Jupyter Notebook. Google Colab надає доступ користувачам до своїх тензорних процесорів (TPU). Тензорний процесор – це спеціалізована інтегральна схема (ASIC), розроблена Google для за дач машинного навчання з використанням бібліотеки TensorFlow. TPU які доступні кожному користувачу мають 64 ГБ пам’яті та продуктивність у 180 TFlops (180 трильйонів операцій в секунду). Отже все що необхідно користувачу для роботи із програмним продуктом це підключення до інтернету, та наявність веб браузера для виходу в мережу Інтернет.

4.3 Архітектура программного забезпечення

4.3.1 Діаграмма класів

На рисунку 4.1 зображена діаграма класів розробленого програмного продукту.

Рисунок 4.1 Діаграмма класів

Даний програмний продукт має чотири класи: DataSet, Plotter, RadarChart, Class_Fit. Dataset – це клас, який містить вхідні дані. У ньому відбувається вся обробка даних: видалення некоректно заповнених елементів, видалення аномалій, перетворення змінних(feature engineering). Plotter – це клас який відповідає за побудову графіків, та діаграм. Серед його атрибутів можна виділити дата сет, та параметри функцій для побудови графіків. Class_Fit – це наша модель. Усі наші алгоритми класифікації є екземплярами цього класу. Class_Fit має наступні атрибути: тестові та тренувальні вибірки, кількість кластерів (якщо алгоритм сам їх не визначає), вид метрики, вид алгоритму та його параметри.

4.3.2 Діаграмма разгортання

На рисунку 4.2 можна побачити діаграму розгортання. Дана діаграма має три компоненти : Web Server, DB Server та User PC.

Рисунок 4.2 Діаграма розгортання.

4.3.3 Діаграмма послідовності

На рисунку 4.3 злбражена діаграми послідовності. На ній показані взаємодії між об’єктами, впорядковані за часом.

Рисунок 4.3 Діаграма послідовності

Висновок до розділу

У даному розділі ми провели аналіз можливих засобів розробки, та аргументували наш вибір. Були визначені загальні вимоги до технічного забезпечення, та зображена архітектура програмного забезпечення.

5 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

5.1 Керівництво користувача

Розглянемо функції, реалізовані у нашому програмному продукті та порівняємо їх із функціональними вимогами до нашої системи, визначеними у першому розділі. Відповідно до розділу 1.3 «Опис функціональної моделі» користувач має виконувати наступні функції:

- завантаження даних;
- вибір інструменту для кластерного аналізу;
  - вибір алгоритму;
  - вибір параметрів алгоритмів;
- виведення графіків та діаграм;
- створити модель.

Усі ці функції були реалізовані при розробці програмного забезпечення. Як користувач взаємодіє із програмою можна побачити на рисунку 5.1

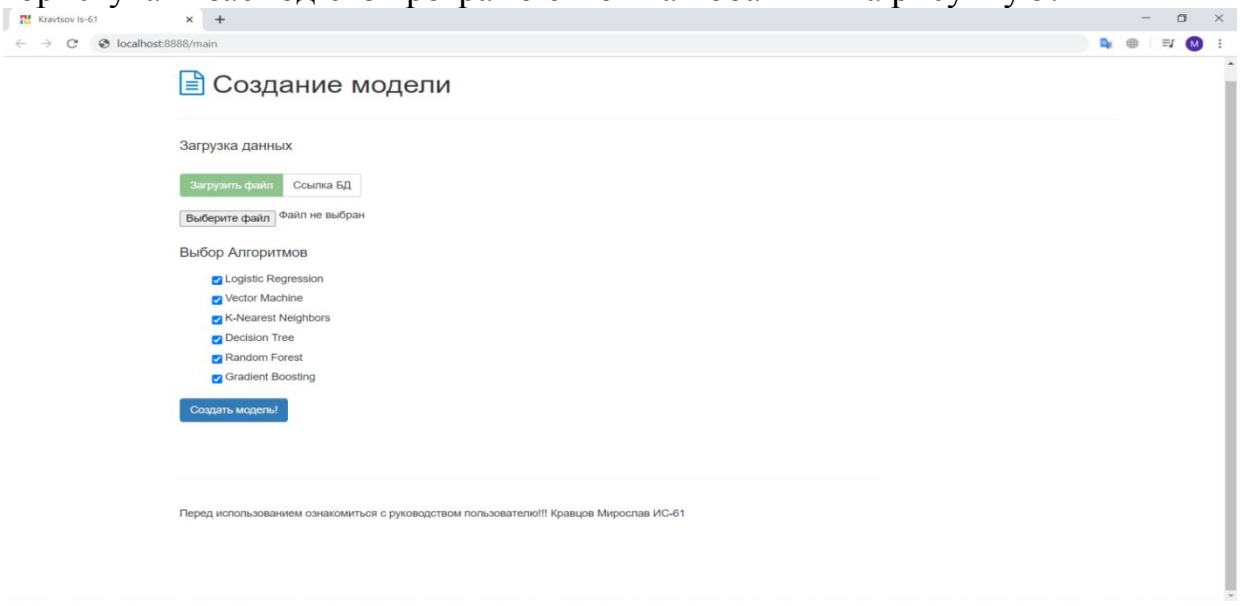

Рисунок 5.1 Головна сторінка

Користувач може підключити вхідні дані двома способами: завантажити csv файл зі свого девайсу, або під'єднатися до SQL бази даних. Слід зауважити що структура вхідних даних має відповідати структурі, яка надана у другому розділі. Таблиця даних має містити наступні поля:

- номер транзакції (InvoiceNo);
- код товару (StockCode);
- опис товару (Description);
- кількість копій цього товару в чеку (Quantity);
- дата покупки (InvoiceDate);
- ціна товару (UnitPrice);
- id покупця (CustomerID);
- країна покупця (Country).

Замість опису товару можна використовувати інформацію про категорію товарів. Після завантаження даних користувачу слід обрати алгоритми, яки будуть виконувати кластерний аналіз. Серед усіх обраних алгоритмів буде обраний алгоритм із найбільшою точністю, саме він буде збережений у файл моделі. Заздалегідь важко виділити найкращий алгоритм, адже на різних даних якість роботи алгоритмів може суттєво відрізнятися. Після обрання алгоритмів користувач має натиснути кнопку створити модель. Після закінчення тренування моделі, звіт у форматі pdf автоматично завантажиться на комп'ютер користувача. Звіт містить графіки (рисунок 2.3, рисунок 2.4, рисунок 3.2, рисунок 3.3, рисунок 3.4) та їх короткий опис.

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань є перевірка відповідності програмного продукту вимогам ТЗ, пошук слабких місць системи, та прийняття заходів по їх усуненню.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Під час тестування були перевірені усі функції які виконує програмний продукт. Результати тестування наведені у таблицях 5.1-5.4

Таблиця 5.1 – Немає вхідних даних

Мета тесту	Перевірка функції «Створити модель!»
Початковий стан	Відкрита сторінка «Створення моделі»
Вхідні дані	Вхідні дані не передбачені
Схема проведення тесту	Вибрати необхідні алгоритми, та натиснути «Створити модель!»
Очікуваний результат	Відкрита сторінка «Створення моделі»
Стан системи після проведення випробувань	Відкрита сторінка «Створення моделі»

Таблиця 5.2 – Файл неправильного формату

Мета тесту	Перевірка функції «Створити модель!»
Початковий стан	Відкрита сторінка «Створення моделі»
Вхідні дані	Завантажений файл хибного формату (має бути csv)
Схема проведення тесту	Вибрати необхідні алгоритми, та натиснути «Створити модель!»
Очікуваний результат	Відкрита сторінка «Створення моделі»
Стан системи після проведення випробувань	Відкрита сторінка «Створення моделі» Виведення повідомлення «Будь ласка завантажте файл у форматі csv»

Таблиця 5.3 – Немає підключення до БД

Мета тесту	Перевірка функції «Створити модель!»
Початковий стан	Відкрита сторінка «Створення моделі»
Вхідні дані	Хибне посилання до БД
Схема проведення тесту	Вибрати необхідні алгоритми, та натиснути «Створити модель!»
Очікуваний результат	Відкрита сторінка «Створення моделі»
Стан системи після проведення випробувань	Відкрита сторінка «Створення моделі» Виведення повідомлення «Будь ласка перевірте посилання до вашої БД»

Таблиця 5.4 – Не обраний жоден алгоритм

Мета тесту	Перевірка функції «Створити модель!»
Початковий стан	Відкрита сторінка «Створення моделі»
Вхідні дані	Вхідні дані завантажені
Схема проведення тесту	Не обирати алгоритми, та натиснути «Створити модель!»
Очікуваний результат	Відкрита сторінка «Створення моделі»
Стан системи після проведення випробувань	Відкрита сторінка «Створення моделі» Виведення повідомлення «Будь ласка оберіть необхідні алгоритми»

Висновок до розділу

У п’ятому розділі ми нписали керівництво для користувача, сформували види випробувань нашого програмного продукту, та навели результати їх проходження. У ході тестувань було встановлено, що розроблена система відповідає функціональним вимогам.

ЗАГАЛЬНІ ВИСНОВКИ

У сучасному світі кожен інтернет магазин має велику кількість даних про покупки його клієнтів. Проаналізувавши ці дані продавець може краще зрозуміти поведінку своїх клієнтів, та можливо навіть навчитися передбачати їх поведінку, що в свою чергу, може привести до збільшення продуктивності роботи магазину, та принести його власнику великі гроші. Проте аналіз таких великих об’ємів даних – це дуже складна задача для мозку людини, і навіть великий штаб аналітиків не зможе виконати це завдання «своїми силами».

На допомогу приходять алгоритми машинного навчання. Для них на відміну від людей великі об’єми інформації це скоріше гарна новина. Залишається лише правильно реалізувати ці алгоритми.
На початку роботи над дипломним проєктом було визначено актуальність данної теми та досліджені існуючі аналоги, які виконують схожі функції. Система дозволяє користувачу обирати необхідні алгоритми для вирішення його задачі, після чого навчає кожну модель окремо та порівнює їх результати. Після чого кластерний аналіз викунуєтсья алгоритмом, який мав найвищу точність.
При виборі алгоритмів та їх розробці слід мати гарне представлення про вхідні дані. Їх структуру та зміст ми проаналізували у другому розділі. Після аналізу даних їх треба обробити. Під час розробки програмного забезпечення ми видалили усі елементи вхідних даних, які були некоректно заповнені. Деякі поля вхідних даних також потребували переробки. Наприклад із поля опису товару (Descriptions) ми зробили поле категорія товару (за допомогою кластеризації). Після цього ми перейшли до розробки алгоритмів.
Було вирішено розробити 6 алгоритмів кластерного аналізу: Suport Vector Machine, Logistic Regression, K-neares Neighbours, Decision Tree, Random Forest, Gradient Boosting.

Точність їх роботи в середньому складала 70%. Найвищу точність показав алгоритм випадкового лісу (Random Forest). Це ансамблевий метод машинного навчання, який складається з баготьох дерев, кожне з яких проводить класифікацію своєї підвибірки (випадкова частина наших вхідних даних). Принадлежність елементу до конкретного класу виявляється шляхом голосування (кожне дерево голосує до якого класу віднести елемент).
У останньому розділі був описаний процес тестування нашого програмного продукту та перевірено його відповідність функціональним вимогам.

Наприкінці хочеться зазначити що точність, яка дорівнює 75 відсоткам – це задовільний результат для поставленої задачі, адже покупці не мають чітко визначених класів, що в свою чергу не дає змогу 100 відсотково їх сегментувати. Також слід розуміти що покупка товарів сильно залежить від пори року під час яких вони зроблені. Наприклад, якщо навчити модель за проміжок часу весна – осінь, та протестувати її на покупках, зроблених на протязі зими, результати скоріж за все будуть погані.

Отже, цілі та вимоги , що були визначені на початковому етапі проєктування системи для проведення кластерного аналізу клієнтського ринка були досягненими в ході виконання дипломного проєкту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Raymond J. Mining Knowledge from Text Using Information Extraction / J. Raymond, B. Razvan. // Department of Computer Sciences, University of Texas at Austin. – С. 1–10.

2. Vandana K. Text classification and classifiers / Korde Sardar Vandana. // National Institute of Technology, Surat.

3. Словник української мови – Київ, 1978. – (Том 9).

4. Daniel Jurafsky. Classification: Naive Bayes, Logistic Regression, Sentiment / Daniel Jurafsky, James H. Martin // Speech and Language Processing / Daniel Jurafsky, James H. Martin., 2015. – С. 1–28.

5. J Korean Acad Nurs. An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain / J Korean Acad Nurs. // College of Nursing and System Biomedical Informatics National Core Research Center, Seoul National University, Seoul, Korea.

6. Manabu Sassano. Virtual Examples for Text Classification with Support Vector Machines / Manabu Sassano. // Fujitsu Laboratories Ltd, Kawasaki 211-8588, Japan.

7. Barry de Ville. Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner / Barry de Ville. – Cary, NC, USA: SAS Institute Inc., 2006.

8. Kevin P. Murphy. Naive Bayes classifier / Kevin P. Murphy. – Vancouver, Canada: Department of Computer Science, University of British Columbia, 2006.

9. Peter M. Lee. Bayesian Statistics: An Introduction, 4th Edition / Peter M. Lee. – 486 с.

10. Mark Lutz. Learning Python / Mark Lutz., 2004. – 552 с.

11. PostgreSQL 10.7 Documentation. // The PostgreSQL Global Development Group. – 2019.

Додаток А

Комплекс задач для аналізу та прогнозування потреб споживачів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

74 арк, 424 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року
Додаток А

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import datetime, nltk, warnings
import matplotlib.cm as cm
import itertools
```

```
from pathlib import Path
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn import preprocessing, model_selection, metrics, feature_selection
from sklearn.model_selection import GridSearchCV, learning_curve
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn import neighbors, linear_model, svm, tree, ensemble
from wordcloud import WordCloud, STOPWORDS
from sklearn.ensemble import AdaBoostClassifier
from sklearn.decomposition import PCA
from IPython.display import display, HTML
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode,iplot
init_notebook_mode(connected=True)
warnings.filterwarnings("ignore")
plt.rcParams["patch.force_edgecolor"] = True
plt.style.use('fivethirtyeight')
mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)
get_ipython().run_line_magic('matplotlib', 'inline')



data = pd.read_csv("data_diplom.csv", encoding= 'unicode_escape')
data.head(10)



df_initial = pd.read_csv('data_diplom.csv',encoding="ISO-8859-1",
                         dtype={'CustomerID': str,'InvoiceID': str})
print('Dataframe dimensions:', df_initial.shape)
#______
df_initial['InvoiceDate'] = pd.to_datetime(df_initial['InvoiceDate'])
#_____________________________________________________________________
# gives some infos on columns types and numer of null values
tab_info=pd.DataFrame(df_initial.dtypes).T.rename(index={0:'column type'})
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()).T.rename(index
={0:'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()/df_initial.shape[
0]*100).T.
```

```
                    rename(index={0:'null values (%)'}))
display(tab_info)
#_____________________
# show first lines
display(df_initial[:5])



df_initial.dropna(axis = 0, subset = ['CustomerID'], inplace = True)
print('Dataframe dimensions:', df_initial.shape)
#_____________________________________________________________________
# gives some infos on columns types and numer of null values
tab_info=pd.DataFrame(df_initial.dtypes).T.rename(index={0:'column type'})
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()).T.rename(index
={0:'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()/df_initial.shape[
0]*100).T.
                    rename(index={0:'null values (%)'}))
display(tab_info)


temp = df_initial[['CustomerID', 'InvoiceNo', 'Country']].groupby(['CustomerID',
'InvoiceNo', 'Country']).count()
temp = temp.reset_index(drop = False)
countries = temp['Country'].value_counts()
print('Nb. de pays dans le dataframe: {}'.format(len(countries)))



data = dict(type='choropleth',
locations = countries.index,
locationmode = 'country names', z = countries,
text = countries.index, colorbar = {'title':'Кількість транз.'},
colorscale=[[0, 'rgb(224,255,255)'],
            [0.01, 'rgb(166,206,227)'], [0.02, 'rgb(31,120,180)'],
            [0.03, 'rgb(178,223,138)'], [0.05, 'rgb(51,160,44)'],
            [0.10, 'rgb(251,154,153)'], [0.20, 'rgb(255,255,0)'],
            [1, 'rgb(227,26,28)']],
reversescale = False)
#_____________________________
layout = dict(title='Number of orders per country',
```

```
geo = dict(showframe = True, projection={'type':'mercator'}))
#_________________
choromap = go.Figure(data = [data], layout = layout)
iplot(choromap, validate=False)



pd.DataFrame([{'products': len(df_initial['StockCode'].value_counts()),
               'transactions': len(df_initial['InvoiceNo'].value_counts()),
               'customers': len(df_initial['CustomerID'].value_counts()),
              }], columns = ['products', 'transactions', 'customers'], index = ['quantity'])



temp = df_initial.groupby(by=['CustomerID', 'InvoiceNo'],
as_index=False)['InvoiceDate'].count()
nb_products_per_basket = temp.rename(columns = {'InvoiceDate':'Number of
products'})
nb_products_per_basket[:10].sort_values('CustomerID')



nb_products_per_basket['order_canceled'] =
nb_products_per_basket['InvoiceNo'].apply(lambda x:int('C' in x))
display(nb_products_per_basket[:5])
#_____________________________________________________________________
_____________________________
n1 = nb_products_per_basket['order_canceled'].sum()
n2 = nb_products_per_basket.shape[0]
print('Number of orders canceled: {}/{} ({:.2f}%) '.format(n1, n2, n1/n2*100))



display(df_initial.sort_values('CustomerID')[:5])



df_check = df_initial[df_initial['Quantity'] < 0][['CustomerID','Quantity',
                                                   'StockCode','Description','UnitPrice']]
for index, col in  df_check.iterrows():
```

```
    if df_initial[(df_initial['CustomerID'] == col[0]) & (df_initial['Quantity'] == -
col[1])
                & (df_initial['Description'] == col[2])].shape[0] == 0:
        print(df_check.loc[index])
        print(15*'-'+'>'+' HYPOTHESIS NOT FULFILLED')
        break



df_cleaned = df_initial.copy(deep = True)
df_cleaned['QuantityCanceled'] = 0

entry_to_remove = [] ; doubtfull_entry = []

for index, col in  df_initial.iterrows():
    if (col['Quantity'] > 0) or col['Description'] == 'Discount': continue
    df_test = df_initial[(df_initial['CustomerID'] == col['CustomerID']) &
                         (df_initial['StockCode']  == col['StockCode']) &
                         (df_initial['InvoiceDate'] < col['InvoiceDate']) &
                         (df_initial['Quantity']   > 0)].copy()
    #_____________________________________

    if (df_test.shape[0] == 0):
        doubtfull_entry.append(index)
    #_____________________________________

    elif (df_test.shape[0] == 1):
        index_order = df_test.index[0]
        df_cleaned.loc[index_order, 'QuantityCanceled'] = -col['Quantity']
        entry_to_remove.append(index)
    #_____________________________________________________________

    elif (df_test.shape[0] > 1):
        df_test.sort_index(axis=0 ,ascending=False, inplace = True)
        for ind, val in df_test.iterrows():
            if val['Quantity'] < -col['Quantity']: continue
            df_cleaned.loc[ind, 'QuantityCanceled'] = -col['Quantity']
            entry_to_remove.append(index)
            break
```

```
print("entry_to_remove: {}".format(len(entry_to_remove)))
print("doubtfull_entry: {}".format(len(doubtfull_entry)))



df_cleaned.drop(entry_to_remove, axis = 0, inplace = True)
df_cleaned.drop(doubtfull_entry, axis = 0, inplace = True)
remaining_entries = df_cleaned[(df_cleaned['Quantity'] < 0) &
(df_cleaned['StockCode'] != 'D')]
print("nb of entries to delete: {}".format(remaining_entries.shape[0]))
remaining_entries[:5]



df_cleaned[(df_cleaned['CustomerID'] == 14048) & (df_cleaned['StockCode'] ==
'22464')]



df_cleaned['TotalPrice'] = df_cleaned['UnitPrice'] * (df_cleaned['Quantity'] -
df_cleaned['QuantityCanceled'])
df_cleaned.sort_values('CustomerID')[:5]

temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'],
as_index=False)['TotalPrice'].sum()
basket_price = temp.rename(columns = {'TotalPrice':'Basket Price'})
#_____________________

df_cleaned['InvoiceDate_int'] = df_cleaned['InvoiceDate'].astype('int64')
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'],
as_index=False)['InvoiceDate_int'].mean()
df_cleaned.drop('InvoiceDate_int', axis = 1, inplace = True)
basket_price.loc[:, 'InvoiceDate'] = pd.to_datetime(temp['InvoiceDate_int'])
#_____________________________________________
basket_price = basket_price[basket_price['Basket Price'] > 0]
basket_price.sort_values('CustomerID')[:6]
```

```
price_range = [0, 50, 100, 200, 500, 1000, 5000, 50000]
count_price = []
for i, price in enumerate(price_range):
    if i == 0: continue
    val = basket_price[(basket_price['Basket Price'] < price) &
                       (basket_price['Basket Price'] > price_range[i-1])]['Basket
Price'].count()
    count_price.append(val)
#_____________________________________________
plt.rc('font', weight='bold')
f, ax = plt.subplots(figsize=(11, 6))
colors = ['yellowgreen', 'gold', 'wheat', 'c', 'violet', 'royalblue','firebrick']
labels = [ '{}<.<{}'.format(price_range[i-1], s) for i,s in enumerate(price_range) if i
!= 0]
sizes  = count_price
explode = [0.0 if sizes[i] < 100 else 0.0 for i in range(len(sizes))]
ax.pie(sizes, explode = explode, labels=labels, colors = colors,
       autopct = lambda x:'{:1.0f}%'.format(x) if x > 1 else '',
       shadow = False, startangle=0)
ax.axis('equal')
f.text(0.5, 1.01, "Mean amount", ha='center', fontsize = 18);


sample_silhouette_values = silhouette_samples(matrix, clusters)
#____________________

graph_component_silhouette(n_clusters, [-0.07, 0.33], len(X),
sample_silhouette_values, clusters)



liste = pd.DataFrame(liste_produits)
liste_words = [word for (word, occurence) in list_products]

occurence = [dict() for _ in range(n_clusters)]

for i in range(n_clusters):
    liste_cluster = liste.loc[clusters == i]
    for word in liste_words:
        if word in ['art', 'set', 'heart', 'pink', 'blue', 'tag']: continue
        occurence[i][word] = sum(liste_cluster.loc[:, 0].str.contains(word.upper()))
```

```
#_____
_______
def random_color_func(word=None, font_size=None, position=None,
                      orientation=None, font_path=None, random_state=None):
    h = int(360.0 * tone / 255.0)
    s = int(100.0 * 255.0 / 255.0)
    l = int(100.0 * float(random_state.randint(70, 120)) / 255.0)
    return "hsl({}, {}%, {}%)".format(h, s, l)
#_____
_______
def make_wordcloud(liste, increment):
    ax1 = fig.add_subplot(4,2,increment)
    words = dict()
    trunc_occurences = liste[0:150]
    for s in trunc_occurences:
        words[s[0]] = s[1]
    #_____
    wordcloud = WordCloud(width=1000,height=400,
background_color='lightgrey',
                          max_words=1628,relative_scaling=1,
                          color_func = random_color_func,
                          normalize_plurals=False)
    wordcloud.generate_from_frequencies(words)
    ax1.imshow(wordcloud, interpolation="bilinear")
    ax1.axis('off')
    plt.title('категорія n°{}'.format(increment-1))
#_____
_______
fig = plt.figure(1, figsize=(14,14))
color = [0, 160, 130, 95, 280, 40, 330, 110, 25]
for i in range(n_clusters):
    list_cluster_occurences = occurence[i]

    tone = color[i] # define the color of the words
    liste = []
    for key, value in list_cluster_occurences.items():
        liste.append([key, value])
    liste.sort(key = lambda x:x[1], reverse = True)
    make_wordcloud(liste, i+1)
```

```
pca = PCA()
pca.fit(matrix)
pca_samples = pca.transform(matrix)



fig, ax = plt.subplots(figsize=(14, 5))
sns.set(font_scale=1)
plt.step(range(matrix.shape[1]), pca.explained_variance_ratio_.cumsum(),
where='mid',
        label='cumulative explained variance')
sns.barplot(np.arange(1,matrix.shape[1]+1), pca.explained_variance_ratio_,
alpha=0.5, color = 'g',
          label='individual explained variance')
plt.xlim(0, 100)

ax.set_xticklabels([s if int(s.get_text())%2 == 0 else '' for s in ax.get_xticklabels()])

plt.ylabel('Explained variance', fontsize = 14)
plt.xlabel('Principal components', fontsize = 14)
plt.legend(loc='upper left', fontsize = 13);


pca = PCA(n_components=50)
matrix_9D = pca.fit_transform(matrix)
mat = pd.DataFrame(matrix_9D)
mat['cluster'] = pd.Series(clusters)


import matplotlib.patches as mpatches

sns.set_style("white")
sns.set_context("notebook", font_scale=1, rc={"lines.linewidth": 2.5})

LABEL_COLOR_MAP = {0:'orange', 1:'red', 2:'b', 3:'purple', 4:'c', 5:'g'}
label_color = [LABEL_COLOR_MAP[l] for l in mat['cluster']]

fig = plt.figure(figsize = (15,8))
increment = 0
for ix in range(4):
```

```
    for iy in range(ix+1, 4):
        increment += 1
        ax = fig.add_subplot(2,3,increment)
        ax.scatter(mat[ix], mat[iy], c= label_color, alpha=0.4)
        plt.ylabel('PCA {}'.format(iy+1), fontsize = 12)
        plt.xlabel('PCA {}'.format(ix+1), fontsize = 12)
        ax.yaxis.grid(color='lightgray', linestyle=':')
        ax.xaxis.grid(color='lightgray', linestyle=':')
        ax.spines['right'].set_visible(False)
        ax.spines['top'].set_visible(False)

        if increment == 9: break
    if increment == 9: break


comp_handler = []
for i in range(5):
    comp_handler.append(mpatches.Patch(color = LABEL_COLOR_MAP[i], label
= i))

plt.legend(handles=comp_handler, bbox_to_anchor=(1.1, 0.97),
           title='Cluster', facecolor = 'lightgrey',
           shadow = True, frameon = True, framealpha = 1,
           fontsize = 13, bbox_transform = plt.gcf().transFigure)

plt.show()




corresp = dict()
for key, val in zip (liste_produits, clusters):
    corresp[key] = val
#_____
_________
df_cleaned['categ_product'] = df_cleaned.loc[:, 'Description'].map(corresp)



for i in range(5):
    col = 'categ_{}'.format(i)
```

```
    df_temp = df_cleaned[df_cleaned['categ_product'] == i]
    price_temp = df_temp['UnitPrice'] * (df_temp['Quantity'] -
df_temp['QuantityCanceled'])
    price_temp = price_temp.apply(lambda x:x if x > 0 else 0)
    df_cleaned.loc[:, col] = price_temp
    df_cleaned[col].fillna(0, inplace = True)
#_____________________________________________________________
_____
df_cleaned[['InvoiceNo', 'Description', 'categ_product', 'categ_0', 'categ_1',
'categ_2', 'categ_3','categ_4']][:5]


temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'],
as_index=False)['TotalPrice'].sum()
basket_price = temp.rename(columns = {'TotalPrice':'Basket Price'})
#_____________________________________________________________

for i in range(5):
    col = 'categ_{}'.format(i)
    temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'],
as_index=False)[col].sum()
    basket_price.loc[:, col] = temp
#_____________________

df_cleaned['InvoiceDate_int'] = df_cleaned['InvoiceDate'].astype('int64')
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'],
as_index=False)['InvoiceDate_int'].mean()
df_cleaned.drop('InvoiceDate_int', axis = 1, inplace = True)
basket_price.loc[:, 'InvoiceDate'] = pd.to_datetime(temp['InvoiceDate_int'])
#_____________________________________________

basket_price = basket_price[basket_price['Basket Price'] > 0]
basket_price.sort_values('CustomerID', ascending = True)[:5]




print(basket_price['InvoiceDate'].min(), '->',  basket_price['InvoiceDate'].max())
```

```
def _scale_data(data, ranges):
    (x1, x2) = ranges[0]
    d = data[0]
    return [(d - y1) / (y2 - y1) * (x2 - x1) + x1 for d, (y1, y2) in zip(data, ranges)]

class RadarChart():
    def __init__(self, fig, location, sizes, variables, ranges, n_ordinate_levels = 6):

        angles = np.arange(0, 360, 360./len(variables))

        ix, iy = location[:] ; size_x, size_y = sizes[:]

        axes = [fig.add_axes([ix, iy, size_x, size_y], polar = True,
        label = "axes{}".format(i)) for i in range(len(variables))]

        _, text = axes[0].set_thetagrids(angles, labels = variables)

        for txt, angle in zip(text, angles):
            if angle > -1 and angle < 181:
                txt.set_rotation(angle - 90)
            else:
                txt.set_rotation(angle - 270)

        for ax in axes[1:]:
            ax.patch.set_visible(False)
            ax.xaxis.set_visible(False)
            ax.grid("off")

        for i, ax in enumerate(axes):
            grid = np.linspace(*ranges[i],num = n_ordinate_levels)
            grid_label = [""]+["{:.0f}".format(x) for x in grid[1:-1]]
            ax.set_rgrids(grid, labels = grid_label, angle = angles[i])
            ax.set_ylim(*ranges[i])

        self.angle = np.deg2rad(np.r_[angles, angles[0]])
        self.ranges = ranges
        self.ax = axes[0]

    def plot(self, data, *args, **kw):
```

```
        sdata = _scale_data(data, self.ranges)
        self.ax.plot(self.angle, np.r_[sdata, sdata[0]], *args, **kw)

    def fill(self, data, *args, **kw):
        sdata = _scale_data(data, self.ranges)
        self.ax.fill(self.angle, np.r_[sdata, sdata[0]], *args, **kw)

    def legend(self, *args, **kw):
        self.ax.legend(*args, **kw)

    def title(self, title, *args, **kw):
        self.ax.text(0.9, 1, title, transform = self.ax.transAxes, *args, **kw)


fig = plt.figure(figsize=(10,12))

attributes = ['count', 'mean', 'sum', 'categ_0', 'categ_1', 'categ_2', 'categ_3',
'categ_4']
ranges = [[0.01, 10], [0.01, 1500], [0.01, 10000], [0.01, 75], [0.01, 75], [0.01, 75],
[0.01, 75], [0.01, 75]]
index  = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

n_groups = n_clusters ; i_cols = 3
i_rows = n_groups//i_cols
size_x, size_y = (1/i_cols), (1/i_rows)

for ind in range(n_clusters):
    ix = ind%3 ; iy = i_rows - ind//3
    pos_x = ix*(size_x + 0.05) ; pos_y = iy*(size_y + 0.05)
    location = [pos_x, pos_y]  ; sizes = [size_x, size_y]
    #_____________________________________________________________
    data = np.array(merged_df.loc[index[ind], attributes])
    radar = RadarChart(fig, location, sizes, attributes, ranges)
    radar.plot(data, color = 'b', linewidth=2.0)
    radar.fill(data, alpha = 0.2, color = 'b')
    radar.title(title = 'cluster n°{}'.format(index[ind]), color = 'r')
    ind += 1


class Class_Fit(object):
    def __init__(self, clf, params=None):
```

```
        if params:
            self.clf = clf(**params)
        else:
            self.clf = clf()

    def train(self, x_train, y_train):
        self.clf.fit(x_train, y_train)

    def predict(self, x):
        return self.clf.predict(x)

    def grid_search(self, parameters, Kfold):
        self.grid = GridSearchCV(estimator = self.clf, param_grid = parameters, cv =
Kfold)

    def grid_fit(self, X, Y):
        self.grid.fit(X, Y)

    def grid_predict(self, X, Y):
        self.predictions = self.grid.predict(X)
        print("Precision: {:.2f} % ".format(100*metrics.accuracy_score(Y,
self.predictions)))
```

```
columns = ['mean', 'categ_0', 'categ_1', 'categ_2', 'categ_3', 'categ_4' ]
X = selected_customers[columns]
Y = selected_customers['cluster']
```

```
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, train_size
= 0.8)
```

```
svc = Class_Fit(clf = svm.LinearSVC)
svc.grid_search(parameters = [{'C':np.logspace(-2,2,10)}], Kfold = 5)
```

```
svc.grid_fit(X = X_train, Y = Y_train)



svc.grid_predict(X_test, Y_test)



def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    #_____________________________________________________
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=0)
    plt.yticks(tick_marks, classes)
    #_____________________________________________________
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    #_____________________________________________________
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
class_names = [i for i in range(11)]
cnf_matrix = confusion_matrix(Y_test, svc.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (8,8))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False,
title='Confusion matrix')



def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=-1, train_sizes=np.linspace(.1, 1.0, 10)):
    """Generate a simple plot of the test and training learning curve"""
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation
score")

    plt.legend(loc="best")
    return plt
```

```
g = plot_learning_curve(svc.grid.best_estimator_,
                        "SVC learning curves", X_train, Y_train, ylim = [1.01, 0.6],
                        cv = 5,  train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5,
                                                0.6, 0.7, 0.8, 0.9, 1])



lr = Class_Fit(clf = linear_model.LogisticRegression)
lr.grid_search(parameters = [{'C':np.logspace(-2,2,20)}], Kfold = 5)
lr.grid_fit(X = X_train, Y = Y_train)
lr.grid_predict(X_test, Y_test)



g = plot_learning_curve(lr.grid.best_estimator_, "Logistic Regression learning
curves", X_train, Y_train,
                        ylim = [1.01, 0.7], cv = 5,
                        train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])



knn = Class_Fit(clf = neighbors.KNeighborsClassifier)
knn.grid_search(parameters = [{'n_neighbors': np.arange(1,50,1)}], Kfold = 5)
knn.grid_fit(X = X_train, Y = Y_train)
knn.grid_predict(X_test, Y_test)



g = plot_learning_curve(knn.grid.best_estimator_, "Nearest Neighbors learning
curves", X_train, Y_train,
                        ylim = [1.01, 0.7], cv = 5,
                        train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])



tr = Class_Fit(clf = tree.DecisionTreeClassifier)
```

```
tr.grid_search(parameters = [{'criterion' : ['entropy', 'gini'], 'max_features' :['sqrt',
'log2']}], Kfold = 5)
tr.grid_fit(X = X_train, Y = Y_train)
tr.grid_predict(X_test, Y_test)
```

```
g = plot_learning_curve(tr.grid.best_estimator_, "Decision tree learning curves",
X_train, Y_train,
                    ylim = [1.01, 0.7], cv = 5,
                    train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

```
rf = Class_Fit(clf = ensemble.RandomForestClassifier)
param_grid = {'criterion' : ['entropy', 'gini'], 'n_estimators' : [20, 40, 60, 80, 100],
          'max_features' :['sqrt', 'log2']}
rf.grid_search(parameters = param_grid, Kfold = 5)
rf.grid_fit(X = X_train, Y = Y_train)
rf.grid_predict(X_test, Y_test)
```

```
g = plot_learning_curve(rf.grid.best_estimator_, "Random Forest learning curves",
X_train, Y_train,
                    ylim = [1.01, 0.7], cv = 5,
                    train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

```
ada = Class_Fit(clf = AdaBoostClassifier)
param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
ada.grid_search(parameters = param_grid, Kfold = 5)
ada.grid_fit(X = X_train, Y = Y_train)
ada.grid_predict(X_test, Y_test)
```

```
g = plot_learning_curve(ada.grid.best_estimator_, "AdaBoost learning curves",
X_train, Y_train,
                    ylim = [1.01, 0.4], cv = 5,
                    train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])


gb = Class_Fit(clf = ensemble.GradientBoostingClassifier)
param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
gb.grid_search(parameters = param_grid, Kfold = 5)
gb.grid_fit(X = X_train, Y = Y_train)
gb.grid_predict(X_test, Y_test)



g = plot_learning_curve(gb.grid.best_estimator_, "Gradient Boosting learning
curves", X_train, Y_train,
                    ylim = [1.01, 0.7], cv = 5,
                    train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])



rf_best  = ensemble.RandomForestClassifier(**rf.grid.best_params_)
gb_best  = ensemble.GradientBoostingClassifier(**gb.grid.best_params_)
svc_best = svm.LinearSVC(**svc.grid.best_params_)
tr_best  = tree.DecisionTreeClassifier(**tr.grid.best_params_)
knn_best = neighbors.KNeighborsClassifier(**knn.grid.best_params_)
lr_best  = linear_model.LogisticRegression(**lr.grid.best_params_)



votingC = ensemble.VotingClassifier(estimators=[('rf', rf_best),('gb', gb_best),
                                    ('knn', knn_best)], voting='soft')



votingC = votingC.fit(X_train, Y_train)
```

```
predictions = votingC.predict(X_test)
print("Precision: {:.2f} % ".format(100*metrics.accuracy_score(Y_test,
predictions)))



basket_price = set_test.copy(deep = True)



transactions_per_user=basket_price.groupby(by=['CustomerID'])['Basket
Price'].agg(['count','min','max','mean','sum'])
for i in range(5):
    col = 'categ_{}'.format(i)
    transactions_per_user.loc[:,col] =
basket_price.groupby(by=['CustomerID'])[col].sum() /
transactions_per_user['sum']*100

transactions_per_user.reset_index(drop = False, inplace = True)
basket_price.groupby(by=['CustomerID'])['categ_0'].sum()

#_____________________________

transactions_per_user['count'] = 5 * transactions_per_user['count']
transactions_per_user['sum']   = transactions_per_user['count'] *
transactions_per_user['mean']

transactions_per_user.sort_values('CustomerID', ascending = True)[:5]



list_cols =
['count','min','max','mean','categ_0','categ_1','categ_2','categ_3','categ_4']
#_____________________________________________________________________
matrix_test = transactions_per_user[list_cols].as_matrix()
scaled_test_matrix = scaler.transform(matrix_test)
```

```
Y = kmeans.predict(scaled_test_matrix)


columns = ['mean', 'categ_0', 'categ_1', 'categ_2', 'categ_3', 'categ_4' ]
X = transactions_per_user[columns]




classifiers = [(svc, 'Support Vector Machine'),
               (lr, 'Logostic Regression'),
               (knn, 'k-Nearest Neighbors'),
               (tr, 'Decision Tree'),
               (rf, 'Random Forest'),
               (gb, 'Gradient Boosting')]
#_____________________________________
for clf, label in classifiers:
    print(30*'_', '\n{}'.format(label))
    clf.grid_predict(X, Y)



predictions = votingC.predict(X)
print("Precision: {:.2f} % ".format(100*metrics.accuracy_score(Y, predictions)))
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проєкту	**В.о. завідувача кафедри**
_____ Олена Клименко___	_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)	(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.	“14” квітня 2020 р.

Комплекс задач для аналізу та прогнозування потреб споживачів

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6114.01.000 ТЗ*

на 8 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Застосування для проведення аналізу ринку та передбачення поведінки покупців.

1.2 Найменування організації-замовника та організацій-учасників робіт

Генеральним замовником проєкту являється кафедра Автоматизованих систем обробки інформації та управління НТУУ "КПІ". Представником замовника є Клименко Олена Миколаївна.

Розробниками системи є студент групи ІС-61 факультету інформатики та обчислювальної техніки НТУУ «КПІ ім. Ігоря Сікорського» Кравцов Мирослав.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проєктно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням системи автоматизації аналізу ринка 5 лютого 2020 рік.

Плановий термін по закінченню роботи над створенням системи автоматизації аналізу ринка – не пізніше 1 червня 2020 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ ЗАСТОСУНКУ

2.1 Призначення застосування

– Застосування призначене допомагати користувачу при аналізі ринку, та прогнозуванні поведінки покупців.

2.2 Цілі створення комплексу задач

– Цілями розробки є:

- полегшити процес аналізу клієнтської бази;
- допомогти користувачу краще зрозуміти своїх покупців;

– Для досягнення поставлених цілей необхідно вирішити наступні задачі:

- проведення кластерного аналізу клієнтів та вивід графіків;
- прогнозування часу через який покупець зробить свою наступну покупку;
- порівняння алгоритмів для кластерного аналізу, та обрання найефективнішого.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для користування сервісом обов'язковою умовою для користувача є наявність браузера, файл із ПЗ можна відкрити на сайті Google Colab, або за допомогою програми Jupyter Notebook. У випадку коли формат ваших даних відповідає вимогам оголошеним у розділі 2 можна використовувати Desktop Application.

Об'єктом автоматизації є проведення кластеризації та аналізу ринку клієнтів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Сервіс має надавати інформацію, у вигляді таблиць та графіків, яка має допомагати користувачу проводити аналіз. Даний сервіс повинен задовольняти потреби користувачів, що цікавляться пошуком інформативних зв'язків у даних, які вони збирають під час роботи їх он-лайн магазинів. Сервіс має виконувати наступні функції:

перевірка формату вхідних даних;

попередня обробка даних;

обрання оптимальних параметрів;

проведення кластеризації товарів та клієнтів;

створення передбачань про подальшу поведінку користувачів.

4.2 Вимоги до надійності

Програма повинна зберігати працездатність і забезпечувати відновлення своїх функцій при виникненні наступних позаштатних ситуацій:

- при помилках в роботі апаратних засобів (крім носіїв даних і програм);
- при помилках, пов'язаних з особливістю різноманітних вхідних даних.

Програмний продукт повинен поєднувати надійність та функціональність. У разі виникнення аварійних ситуацій необхідно сповіщати користувача та надавати інструкцію для подальших дій. Будь-які аварійні ситуації мають бути задокументовані у звіті, який при необхідності надсилається розробнику для визначення причини збою в роботі та усуненні помилок, які могли привести до нестабільної роботи програмного продукту.

4.3 Вимоги до складу і параметрів технічних засобів

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проєктування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити виконання встановлених операцій процесу технічного обслуговування.

Для правильної роботи розробленої системи до складу технічних засобів повинен входити комп'ютер, що має конфігурацію наведену нижче:

а) комп'ютер з такою конфігурацією:

1) процесор з тактовою частотою не нижче 1 ГГц;
2) об'єм оперативної пам'яті не менше 128 Мб;
3) інші складові можуть мати будь-які параметри, тому що вони не значним чином впливають на роботу застосування;
4) підключення до мережі інтернет;

б) додатково має бути встановлене таке програмне забезпечення:

1) процесор з тактовою частотою не нижче 1 ГГц;
2) об'єм оперативної пам'яті не менше 256 Мб;
3) інші складові можуть мати будь-які параметри, тому що вони не значним чином впливають на роботу програми;
4) наявність швидкісного інтернет-з'єднання;

в) комп'ютерна периферія, до складу якої входить:

1) монітор;
2) мишка;
3) клавіатура.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки системи ведення наукової роботи.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1	Підготовка технічного завдання на розробку програмного продукту	07.03.2020	
2	Розробка сценарію роботи	12.03.2020	
3	Технічне проєктування – функціональність, модулі, задачі, цілі тощо	20.03.2020	
4	Узгодження з керівником інтерфейсу користувача	02.04.2020	
5	Розробка інформаційного забезпечення	17.04.2020	
6	Розробка програмного забезпечення	29.04.2020	
7	Налагодження програми	09.05.2020	
8	Тестування програми	13.05.2020	
9	Здача готового програмного продукту замовнику	17.05.2020	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Тестування ефективності роботи алгоритмів відбувається разом із розробкою та налаштування цих алгоритмів (вхідні дані діляться на train set та test set).

Тестування авторизації користувача полягає в представленні користувачу головної сторінки з відповідним повідомленням.

Тестування редагування профілю користувача полягає в представленні користувачу сторінки з відповідною формою та у використанні в подальшому нових даних, введених користувачем.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003979436

Дата перевірки:
12.06.2020 03:31:41 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2020 13:50:17 EEST

ID користувача:
77149

Назва документу: **Kravtsov_is61**

ID файлу: **1003994466** Кількість сторінок: **31** Кількість слів: **4840** Кількість символів: **40437** Розмір файлу: **506.00 KB**

13.3% Схожість

Найбільша схожість: **8.24%** з джерело бібліотеки. ID файлу: **12191126**

5.76% Схожість з Інтернет джерелами	68	Page 33
13.2% Текстові збіги по Бібліотеці акаунту	156	Page 34

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів	2

Графічний матеріал
до дипломного проєкту

на тему: Комплекс задач для аналізу та прогнозування потреб споживачів

Київ – 2020 року

ДП 6114.02.000 ССВ

					ДП 6114.02.000 ССВ
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань
Розробив		Кравцов М.В.			Літера: — Маса: — Масштаб: —
					Аркуш 1 / Аркушів 1
Перевірив		Клименко О.М			
Т. кон.					Комплекс задач для аналізу та прогнозування потреб споживачів
Н. кон.		Теслишева Т.О			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Клименко О.М			

ДП 6114.03.000 ССК

					ДП 6114.03.000 ССК		
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Кравцов М.В.			Схема структурна класів програмного забезпечення	Літера	Маса / Масштаб
Перевірив		Клименко О.М				Аркуш 1	Аркушів 1
Т. кон.					Комплекс задач для аналізу та прогнозування потреб споживачів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О					
Затвердив		Клименко О.М					

ДП 6114.05.000 ССП

					ДП 6114.05.000 ССП
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності
Розробив		Кравцов М.В.			Літера: — Маса: — Масштаб: —
					Аркуш 1 / Аркушів 1
Перевірив		Клименко О.М			
Т. кон.					Комплекс задач для аналізу та прогнозування потреб споживачів
Н. кон.		Тєлишева Т.О			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Клименко О.М			

ДП 6114.04.000 ССР

					ДП 6114.04.000 ССР
Зм.	Арк.	№ документа	Підпис	Дата	Літера / Маса / Масштаб
Розробив		Кравцов М.В.			Схема структурна розгортання програмного забезпечення
Перевірив		Клименко О.М			Аркуш 1 / Аркушів 1
Т. кон.					Комплекс задач для аналізу та прогнозування потреб споживачів
Н. кон.		Тєлишева Т.О			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Клименко О.М			